**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА  
СЕРВИСА ИГРОВЫХ СОХРАНЕНИЙ «PlaySaveBack»**

***ДИПЛОМНЫЙ ПРОЕКТ***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнил:

Студент группы 1912с

\_\_\_\_\_\_\_\_\_\_\_ / Заикин В.А

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Проверил:

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ / Уляхин В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Оглавление

[Введение 3](#_Toc191732807)

[Информация о проекте 3](#_Toc191732808)

[Актуальность 3](#_Toc191732809)

[Глоссарий 4](#_Toc191732810)

[Глава 1. Техническое задание 5](#_Toc191732811)

[Общее описание 5](#_Toc191732812)

[Требования к функциональности 6](#_Toc191732813)

[Ролевая модель 7](#_Toc191732814)

[Диаграмма прецедентов 8](#_Toc191732815)

[Диаграммы последовательностей 9](#_Toc191732816)

[Технологический стек 13](#_Toc191732817)

[Хранимые данные 13](#_Toc191732818)

[Требования к REST API 14](#_Toc191732819)

[Требования к десктопному приложению клиента | Схематические макеты интерфейса 27](#_Toc191732820)

[Глава 2. Проектирование и разработка базы данных 39](#_Toc191732821)

[Концептуальная модель данных 39](#_Toc191732822)

[Логическая модель данных 40](#_Toc191732823)

[Физическая модель данных 41](#_Toc191732824)

# Введение

## Информация о проекте

|  |  |
| --- | --- |
| **Название** | Сервис игровых сохранений «PlaySaveBack» |
| **Цель** | Спроектировать и разработать сервис игровых сохранений «PlaySaveBack» |
| **Задачи** | 1. Разработать базу данных сервиса на MySQL. 2. Спроектировать структуру API сервиса. 3. Разработать сервер API сервиса на фреймворке Laravel. 4. Разработать десктопное приложение на WinUI 3 для клиента. 5. Разработать автоматизированные тесты и провести тестирование. |

## Актуальность

Современные игроки сталкиваются с проблемой хранения и управления своими игровыми сохранениями. Многие игры не предоставляют удобных средств для резервного копирования, версионности или переноса сохранений между устройствами. Это особенно актуально для пользователей, которые переустанавливают операционную систему, используют несколько игровых платформ или хотят вернуться к определённому состоянию игры. Существующие решения, такие как Steam Cloud, не всегда охватывают все игры и предоставляют ограниченные возможности управления резервными копиями.

Разрабатываемая система решает эту проблему, предоставляя удобный инструмент для хранения, управления и синхронизации игровых сохранений в облаке. Пользователи смогут загружать и восстанавливать сохранения, отслеживать изменения, а также делиться своими данными с другими игроками.

Таким образом, система повышает удобство использования сохранений, минимизирует риск потери прогресса и предоставляет новые возможности для взаимодействия пользователей с их игровыми данными.

## Глоссарий

|  |  |
| --- | --- |
| **Понятие** | **Определение понятия** |
| API | (интерфейс программирования приложений) – набор определенных правил и соглашений, который используется для взаимодействия между различными программными компонентами. API определяет, как различные части программного обеспечения могут взаимодействовать друг с другом, обычно путем обмена данными и выполнения определенных операций. |
| Postman | Инструмент для тестирования и разработки API. Он предоставляет удобный интерфейс для отправки HTTP запросов к API, а также для анализа ответов. |
| Draw.io | Бесплатный онлайн и оффлайн инструмент для создания диаграмм и схем. Он предоставляет широкий набор инструментов и элементов, которые позволяют пользователям создавать различные типы диаграмм, такие как организационные диаграммы, блок-схемы, ER-диаграммы, сетевые диаграммы, диаграммы потока процессов и многое другое. |
| CRUD | Акроним, обозначающий четыре базовые функции, используемые при работе с данными: создание, чтение, модификация, удаление. |
| Laravel | Бесплатный, открытый фреймворк для создания веб-приложений на языке программирования PHP. Он предоставляет структуру и базовые инструменты для быстрого создания веб-приложений, используя принципы модели-представления-контроллер (MVC) и другие современные подходы к разработке. |
| CRON | Планировщик задач Unix-подобных ОС, который позволяет автоматически запускать программы или скрипты в заданное время или через определённые интервалы. |
| CORS | Cross-origin resource sharing — технология современных браузеров, которая позволяет предоставить веб-страницам доступ к ресурсам другого домена. |

# Глава 1. Техническое задание

## Общее описание

Разрабатываемый сервис представляет собой систему управления сохранениями видеоигр с возможностью резервного копирования и восстановления данных. В основе проекта лежит клиентское приложение на WinUI 3, взаимодействующее с серверной частью, реализованной на Laravel. Система предназначена для геймеров, которые хотят сохранять, управлять и восстанавливать свои игровые данные без привязки к конкретному устройству.

Игры автоматически загружаются из открытого API Steam. Каждый день в полночь отправляется запрос с помощью CRON, который сверяет список игр с актуальными данными Steam, добавляя новые при их отсутствии в БД.

Система поддерживает версионность сохранений, интеграцию с облачными сервисами и возможность совместного использования сохранений с другими пользователями.

При отсутствии игры в базе данных пользователю предоставляется возможность добавить её в библиотеку в качестве персональной. Это позволит сохранять и управлять файлами сохранений для данной игры. Однако оформление сохранений в виде публичных постов будет доступно только для игр, присутствующих в базе данных.

Функционал гостя охватывает регистрацию, авторизацию и восстановления аккаунта (см. Рисунок 1).

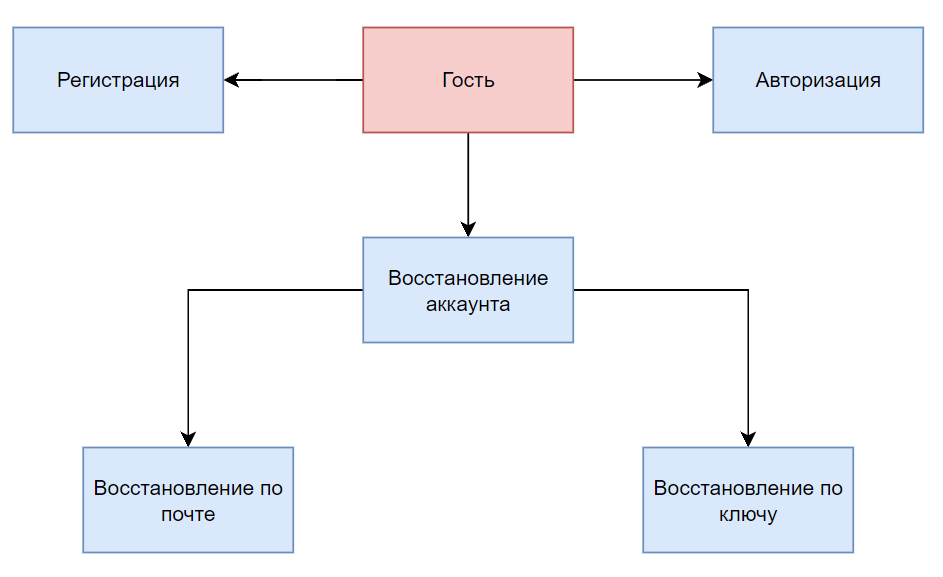


Рисунок 1 – Функционал гостя

**Функционал пользователя** охватывает управление библиотекой игр, работой с сохранениями и профилем, управление своими игровыми данными. Детали функционала пользователя представлены на Рисунок 2.

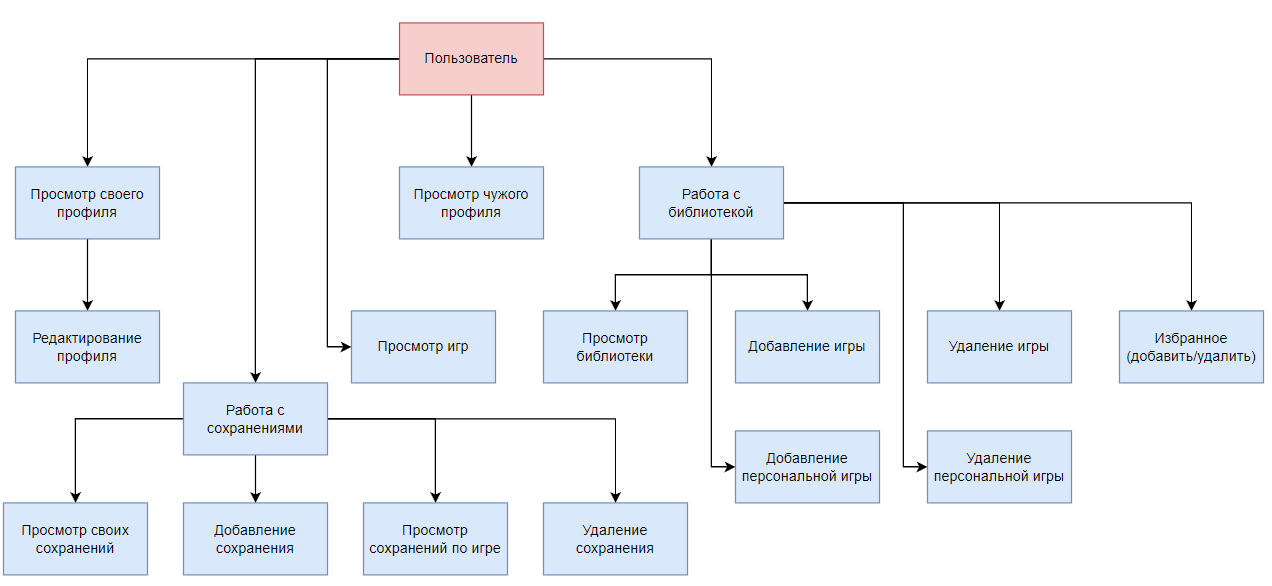


Рисунок 2 – Функционал пользователя

## Требования к функциональности

### Гости

1. Аутентификация и регистрация.
   * Аутентификация производится по токену, который генерируется при входе и отзывается при выходе.
   * Вход на другом устройстве не должен вызывать выход из аккаунта на всех устройствах одного пользователя.
2. Восстановление аккаунта
   * Возможность восстановления аккаунта по e-mail, либо по особому ключу, выданного при регистрации.

### Пользователи

1. Уметь всё что гость.
2. Управление своей библиотекой игр

* Добавление игры в библиотеку из списка доступных игр.
* Добавление персональной игры, если нужной игры нет в БД.
* Удаление игры из библиотеки.
* Добавление или удаление игры в избранное.

1. Управление сохранениями

* Просмотр списка всех сохранений.
* Просмотр сохранений для конкретной игры.
* Добавление нового сохранения для игры.
* Удаление сохранения.
* Создание публичного поста на основе сохранения (только для игр из БД).

1. Работа с профилем

* Просмотр своего профиля.
* Обновление данных профиля.
* Просмотр чужого профиля.

1. Просмотр информации об играх

* Просмотр списка всех доступных игр.
* Просмотр информации о конкретной игре.

### Планировщик задач

1. Автоматическое обновление списка игр:
   * Каждый день в полночь система отправляет запрос к открытому API Steam, получает JSON и сверяет его с предыдущем.
   * Если игры отсутствуют в БД, они добавляются автоматически.

## Ролевая модель

Клиент — основной авторизированный пользователь сервиса, который загружает, организует и пользуется сохранениями.

Гость — неавторизированный пользователь, которому доступна только регистрация, вход и восстановление аккаунта.

Более структурированная и подробная информация представлена в Таблица 1.

Таблица 1 – Ролевая модель системы

|  |  |  |  |
| --- | --- | --- | --- |
| **Раздел** | **Фичи** | **Пользователь** | **Гость** |
| Аутентификация и регистрация | Вход | нет | да |
| Регистрация | нет | да |
| Восстановление аккаунта | нет | да |
| Выход | да | нет |
| Профиль | Просмотр своего профиля | да | нет |
| Просмотр чужого профиля | да | нет |
| Обновление профиля | да | нет |
| Библиотека | Получить свою библиотеку | да | нет |
| Добавить игру в библиотеку | да | нет |
| Удалить игру из библиотеки | да | нет |
| Добавить игру в избранное | да | нет |
| Убрать игру из избранного | да | нет |
| Сохранения | Просмотр всех своих сохранений | да | нет |
| Просмотр всех сохранений к игре | да | нет |
| Просмотр чужих сохранений | да (если публичные) | нет |
| Просмотр сохранений в виде поста | да | нет |
| Добавление сохранения | да | нет |
| Удаления сохранения | да | нет |
| Игры | Просмотр всех игр | да | нет |
| Просмотр игры | да | нет |
| Добавление персональной игры | да | нет |
| Удаление персональной игры | да | нет |

## Диаграмма прецедентов

Диаграмма прецедентов используется для моделирования функциональных требований системы и взаимодействия между пользователями и системой. Она помогает определить основные сценарии использования системы и её ключевые функции. Состоит из следующих ключевых элементов:

* Акторы: роли пользователей или внешних систем, взаимодействующих с системой
* Прецеденты: функциональные возможности или сценарии, которые система предоставляет
* Связи: взаимодействия между акторами и прецедентами
* Включение и расширение: отношения между прецедентами для повторного использования или добавления дополнительного поведения

На диаграмме отражены ключевые функции, такие как управление сохранениями, библиотекой, работа со своим профилем и играми (см. Рисунок 3).

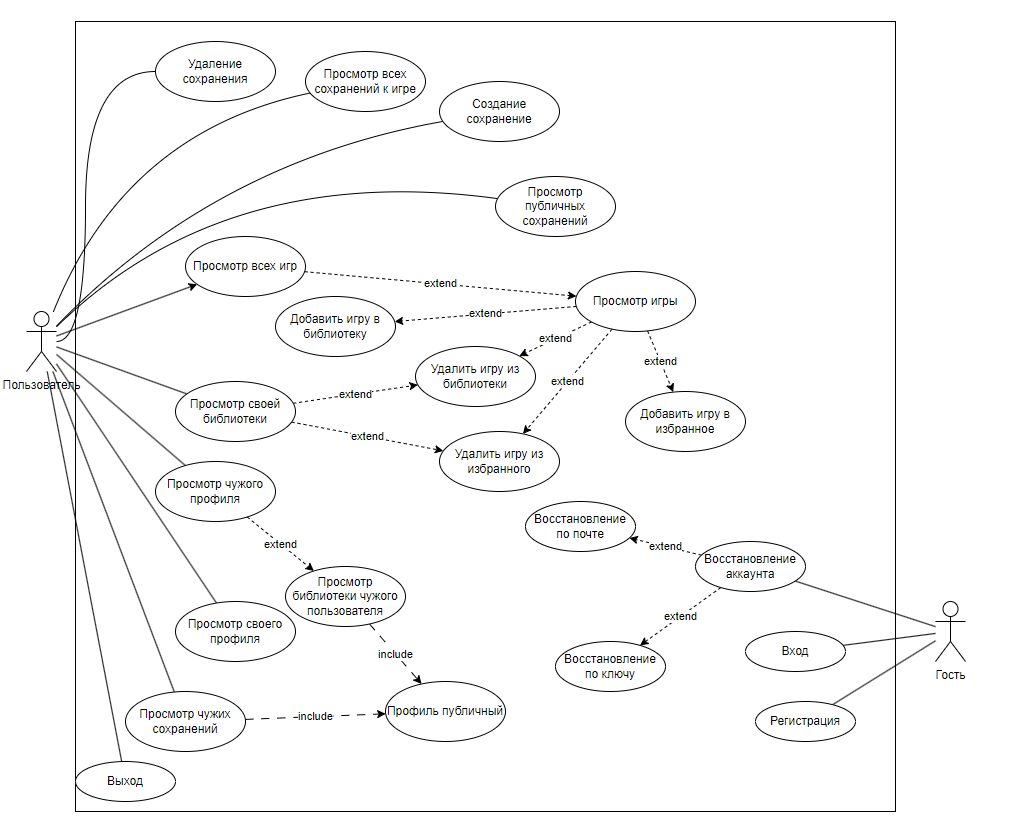


Рисунок 3 – Диаграмма прецедентов

## Диаграммы последовательностей

Диаграмма последовательностей визуализирует взаимодействие объектов в системе в рамках конкретного сценария. Она отражает последовательность сообщений, передаваемых между объектами, и их временную упорядоченность. Состоит из следующих ключевых элементов:

* Объекты и акторы: участники взаимодействия
* Сообщения: вызовы методов или передача данных между объектами
* Временная ось: вертикальная линия, показывающая порядок событий
* Активация: период активности объекта при выполнении операции
* Фреймы: обёртки над элементами для описания конкретного случая или цикла

### Общие диаграммы-дополнения

Общие диаграммы последовательности не являются самостоятельными и предназначены для дополнения к другим последовательностям для разгрузки повторяющихся элементов. Обычные диаграммы ссылаются на дополнительны посредством цветной активации по цвету выхода и припиской «(Общее: <название>)».

Валидация тела POST запроса осуществляется посредником, проверяя заданные правила на ключах (см. Рисунок 4). Имя ссылки: «валид».

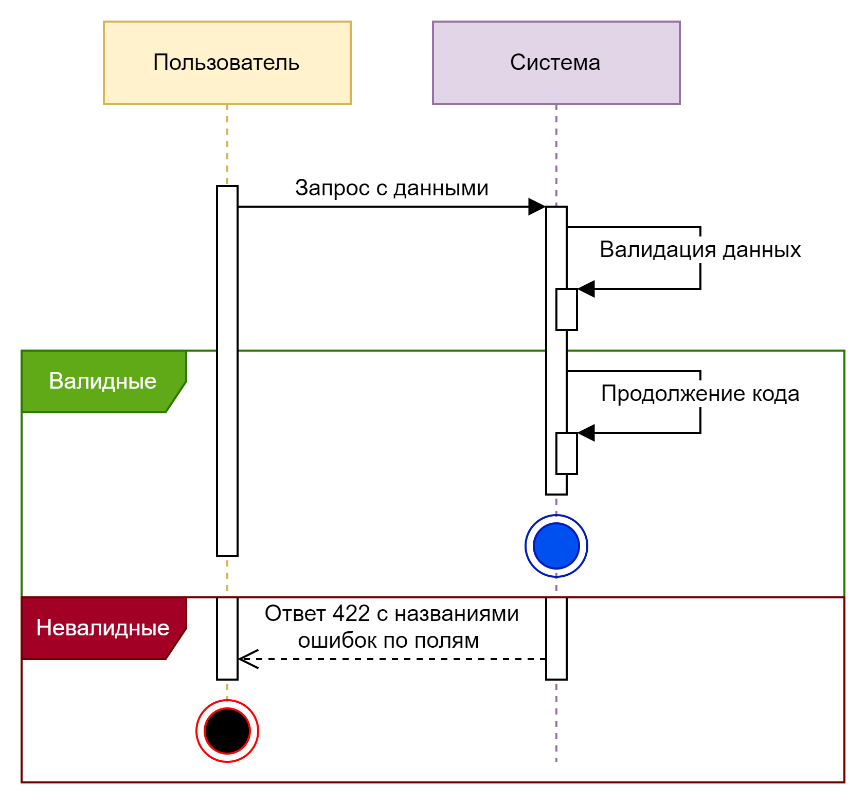


Рисунок 4 – Валидация тела запроса

Внесения в контекст кода модель пользователя и отброс неавторизированных осуществляется посредником, проверяя токен в заголовках запроса и запрашивая пользователя с этим ИД (см. Рисунок 5). Имя ссылки: «токен».

Внесения в контекст кода модель объекта по ИД и отброс если не модель не найдена осуществляется посредником, беря из маршрута ИД и запрашивая из БД её модель (см. Рисунок 6). Имя ссылки: «сущ».

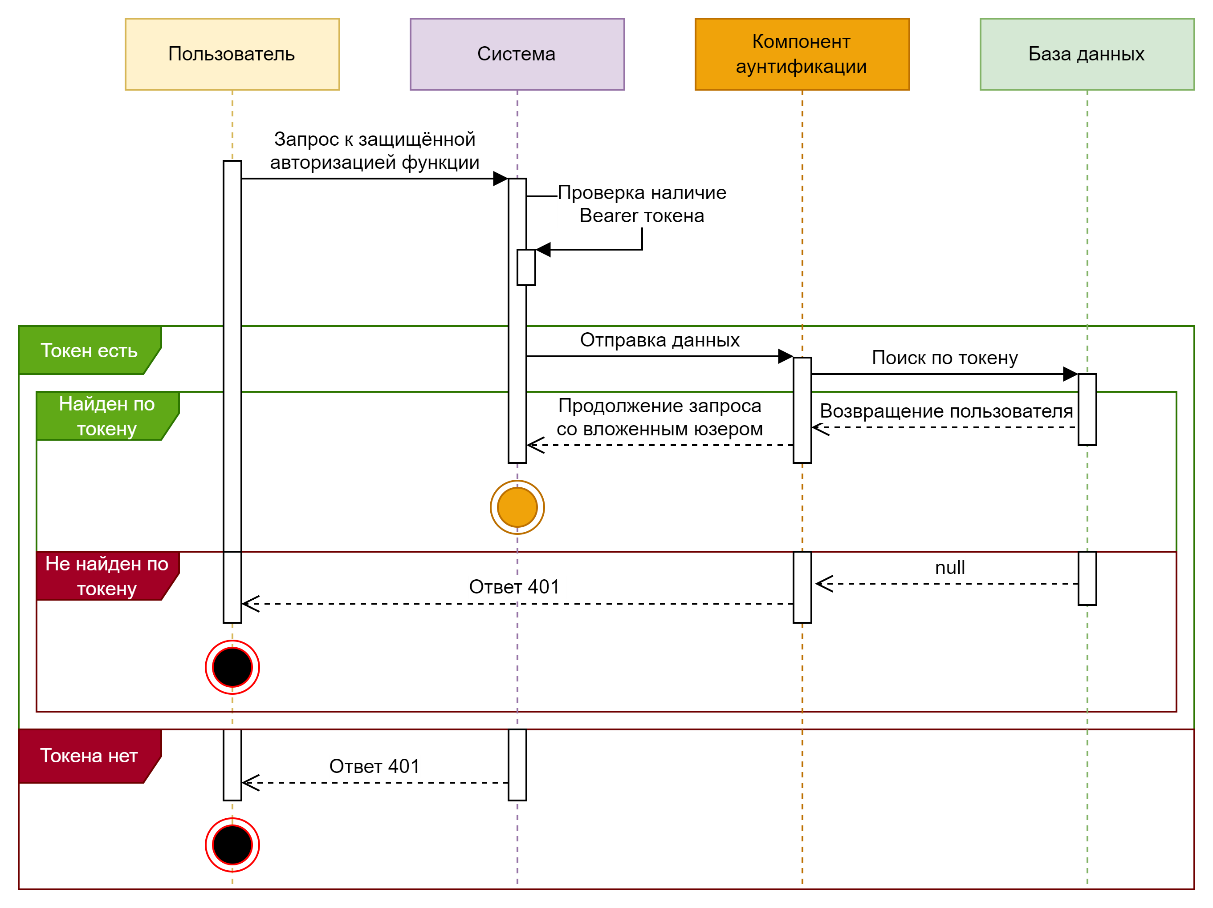


Рисунок 5 – Проверка токена и внесение в контекст пользователя

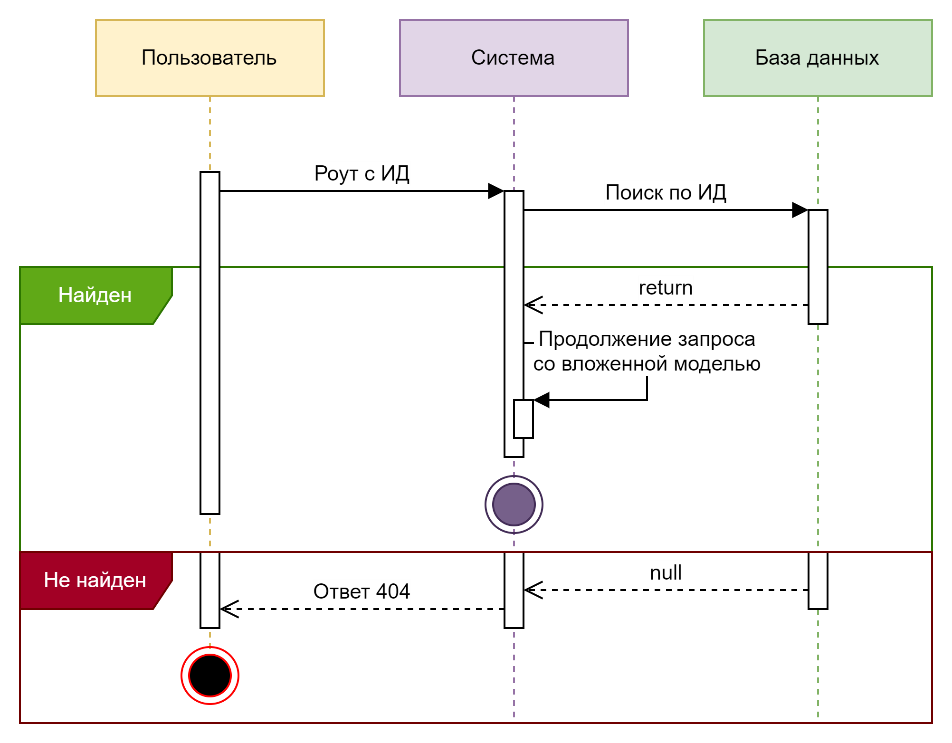


Рисунок 6 – Проверка существования объекта БД

### Авторизационные функции

Регистрация новой учётной записи и последующая авторизация представлена на Рисунок 7.

Вход по реквизитам представлена на Рисунок 8.

Выход и удаление токена представлено на Рисунок 9.

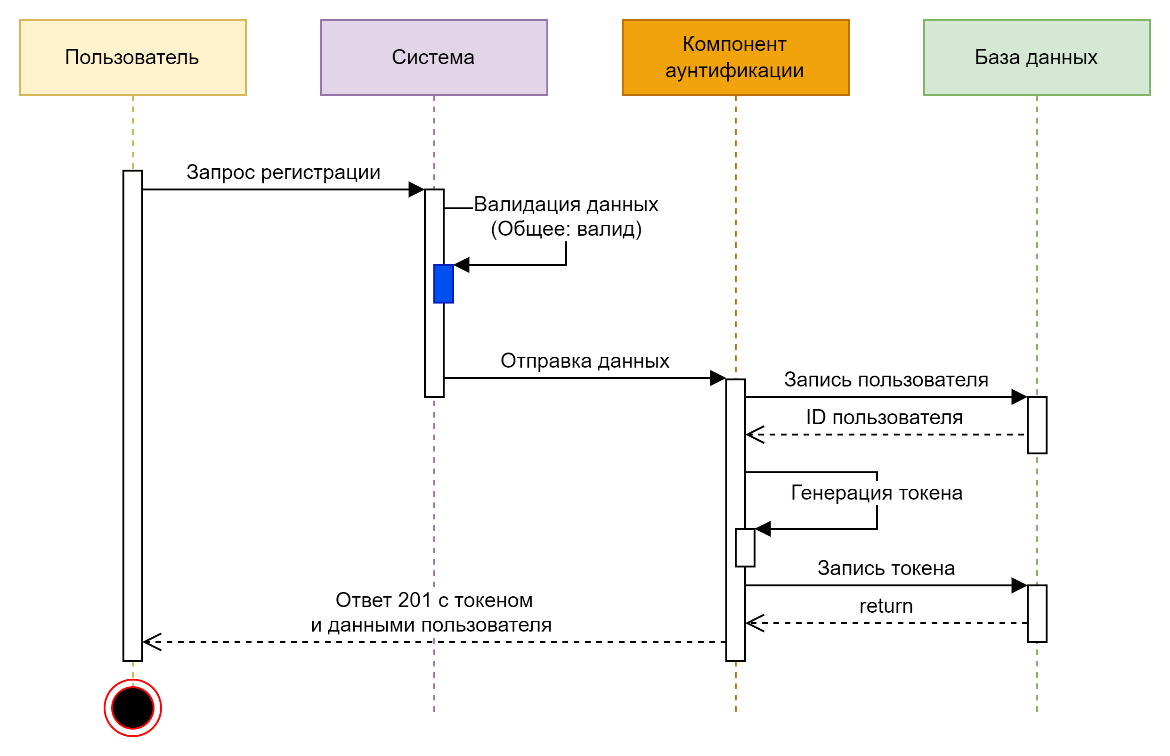


Рисунок 7 – Регистрация

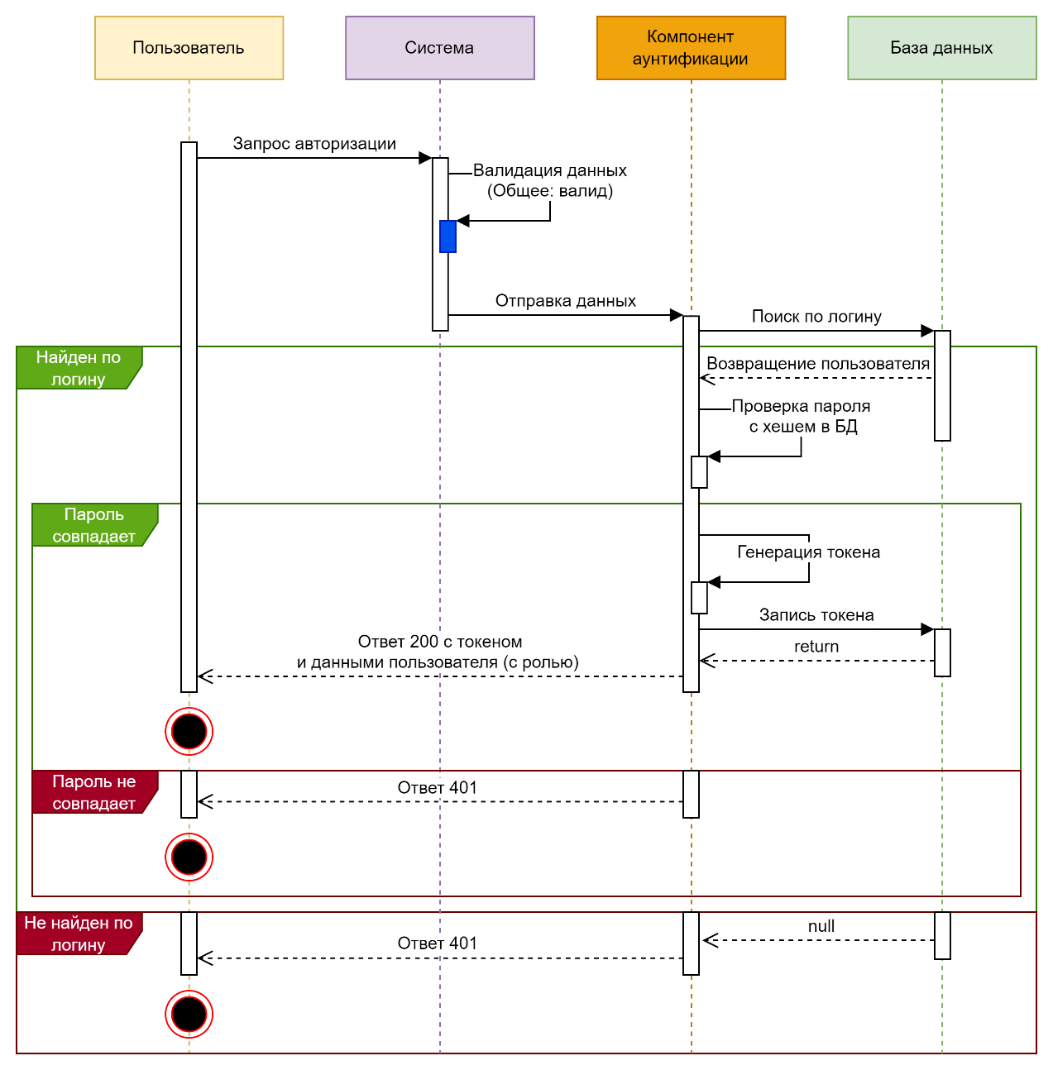


Рисунок 8 – Вход

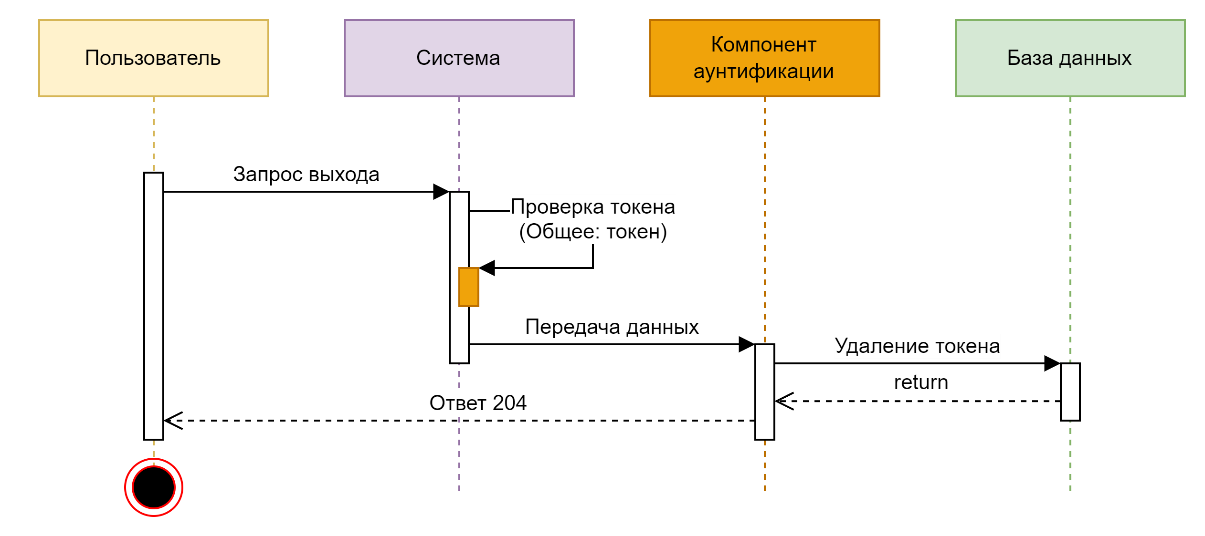


Рисунок 9 – Выход

## Технологический стек

**Серверная часть**

Язык программирования: PHP 8.3

Фреймворк для разработки: Laravel 11

Система управление базами данными: MySQL

**Десктопное приложение**

Язык программирования и разметки: C#, XAML

Фреймворк для разработки: WinUI 3

## Хранимые данные

1. **Игры:** уникальный идентификатор игры, название игры, платформа, код игры.
2. **Пользователи:** уникальный идентификатор пользователя, электронная почта, хеш пароля, никнейм, аватар, логин, заголовок профиля, ключ.
3. **Сохранения:** уникальный идентификатор сохранения, имя файла сохранения, путь к файлу сохранения, размер файла, описание сохранения, идентификатор пользователя, идентификатор игры, идентификатор облачного сервиса пользователя, скриншоты, идентификатор пользовательской игры, тип сохранения.
4. **Облачные сервисы:** уникальный идентификатор облачного сервиса, название облачного сервиса, описание сервиса, иконка сервиса.
5. **Облачные сервисы пользователей:** уникальный идентификатор связи пользователя и облачного сервиса, токен доступа, токен обновления, время истечения токена, идентификатор облачного сервиса, идентификатор пользователя.
6. **Доступы к сохранениям:** уникальный идентификатор доступа, идентификатор сохранения, идентификатор пользователя, тип доступа, время истечения доступа, токен доступа, флаг публичного доступа.
7. **Библиотеки:** уникальный идентификатор библиотеки, идентификатор пользователя, идентификатор игры, время последнего запуска игры, время, проведённое в игре, флаг избранного, идентификатор пользовательской игры.
8. **Пользовательские игры:** уникальный идентификатор пользовательской игры, название пользовательской игры, идентификатор пользователя.

## Требования к REST API

### Общие требования

Должно быть открытое АPI — правила CORS должны разрешать доступ с других доменов.

Идентификацию пользователя для не гостевых функции должно быть организовано посредством Bearer Token.

При попытке доступа к защищенным авторизацией функциям системы во всех запросах необходимо возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 401 Unauthorized |

При попытке доступа авторизованным пользователем к функциям недоступным для своей роли во всех запросах необходимо возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 403 Forbidden |

При попытке получить несуществующий ресурс возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 404 Not Found |

В случае ошибок связанных с валидацией данных в теле запроса необходимо возвращать следующий ответ:

|  |
| --- |
| **Response** |
| **Status:** 422 Unprocessable Content  **Content-Type:** application/json  **Body:** {  "errors": [  <ключ>: [  <ошибка>,  <ошибка>,  ...  ],  <ключ>: [...],  ...  ]  } |

Вместо поля <ключ> должно быть название ключа запроса, в котором произошла ошибка валидации, а вместо <ошибка> — описание допущенной ошибки в этом ключе. Ошибок в одном ключе может быть множество, как и в одном запросе — множество ключей с ошибками.

### Авторизация

#### Регистрация

Запрос позволяет пользователю зарегистрироваться и вернуть его токен.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/register  **Method:** POST  **Content-Type:** application/json  **Body:** {  "name": **{nickname}**,  "login": **{login}**,  "password": **{password}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "token": **<токен>**  "user": <**данные\_пользователя**>  }  **Возможные статус-коды:** 201, 422 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | api | string | да |  | URL до API | https://api.pics.com |
| 2 | nickname | string | да | мин: 2, макс: 255 | Имя юзера | Testello |
| 3 | login | string | да | мин: 2, макс: 64, уник,  regex: ^[a-zA-Z0-9\_-]+$ | Логин | test |
| 4 | password | string | да | мин: 8, макс: 255 | Пароль | Test123! |

#### Авторизация

Запрос позволяет авторизировать пользователя и получить токен доступа.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/login  **Method:** POST  **Content-Type:** application/json  **Body:** {  "login": **{login}**  "password": **{password}**  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "token":<**токен**>  "user": {  <**"role":** **"admin"**>  "name": "Testello",  "login": "test"  }  }  **Возможные статус-коды:** 200, 401, 422 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | login | string | да | Логин | mylogin |
| 2 | password | string | да | Пароль | Pass123! |

#### Выход

Запрос предназначен для удаления текущего токена пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/logout  **Method:** POST  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 200, 401 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | token | string | да | API токен | 27|fxR7b3H3o5nPYAdtWhbk61do |

### Работа с сохранениями

#### Создание альбома

Запрос для создания нового личного альбома пользователя. Возвращает объект альбома с его ID в БД и создаёт папку с выведенным названием в ФС.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "**name**": **{name}**,  "path": **{path}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "id": <**id\_альбома**>,  "name": **{name}**,  "path": **{path}**  }  **Возможные статус-коды:** 201, 401, 409, 422 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | name | string | да | макс: 255, уник(для юзера) | Название | Камера |
| 2 | path | string | нет | макс: 255, уник(для юзера) | Путь на клиенте для синхронизации | DCIM/Camera |

#### Получение альбома

Запрос для удаления альбома. Удаляет все связанные данные в БД и все картинки в ФС.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:**  Свой  {  "album": {  "id": 1,  "name": "Камера",  "createdAt": "2024-12-16T10:13:23",  "grantAccesses": [  {  "id": 215,  "name": "Чел"  }, { ... }, { ... }  ],  "picturesCount": 31,  "picturesInfo": {  "sign": "2\_JDJ5JDEyJG11RGc24aWRX",  "ids": [84, 30, 69, 59]  }  }  }  Чужой доступный  {  "album": {  "id": 1,  "name": "Скаченное",  "owner": {  "id": 215,  "name": "Чел"  },  "picturesCount": 29,  "picturesInfo": {  "sign": "2\_JDJ5JDEyJHoMVpWaEJjVzNSV2",  "ids": [49, 41, 30, 87]  }  }  }  **Возможные статус-коды:** 200, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | albumId | integer | да | существовать | Код альбома | 87 |

#### Удаление альбома

Запрос для удаления альбома. Удаляет все связанные данные в БД и все картинки в ФС.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  **Method:** DELETE  **Authorization:** Bearer **{token}** | **Status:** 204 Deleted  **Возможные статус-коды:** 204, 401, 403, 404 |

#### Получение списка альбомов

Запрос для получения списка альбомов. Возвращает список всех альбомов пользователя и доступных из БД.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:**  {  "own": [  {  "id": 354,  "name": "Камера",  "path": "DCIM/Camera",  "createdAt": "2024-12-16T10:13:23",  "grantAccessesCount": 3,  "picturesCount": 31,  "picturesInfo": { ... }  }, { ... }, { ... }  ],  "accessible": [  {  "id": 5372,  "name": "Скаченное",  "path": "Pictures/Downloads",  "createdAt": "2024-12-16T10:13:23",  "owner": {  "id": 215,  "name": "Чел"  },  "picturesCount": 31,  "picturesInfo": { ... }  }, { ... }, { ... }  ]  }  **Возможные статус-коды:** 200, 401 |

### Работа с доступами к сохранению

#### Создание кода приглашения

Запрос для создания кода приглашения для доступа к альбому. Если лимиты не указаны, то приглашение бессрочное и вступить может любое количество человек.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**/invite  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "expiresAt": **{expiresAt},**  "timeLimit": **{timeLimit}**,  "joinLimit": **{joinLimit}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "invitation": {  "code": "928Lm8e4",  "expiresAt": "2025-12-30 20:50:00",  "joinLimit": 1  }  }  **Возможные статус-коды:** 201, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | expiresAt | datetime | нет | > текущего | Дата просрочки | 2024-12-30 |
| 2 | timeLimit | integer | нет | мин: 1 | Лимит времени (минуты) | 60 |
| 3 | joinLimit | integer | нет | мин: 1 | Лимит вступлений | 1 |

#### Получение альбома в приглашении

Запрос для получения информации об альбоме по коду приглашения. Показывает 30 последних картинок.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/invitation/**{code}**/album  **Method:** GET | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "album": {  "id": 3,  "name": "Огурец",  },  "pictures": [  {...}, {...}, {...}  ]  }  **Возможные статус-коды:** 200, 404 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | code | string | да | Код приглашения | Bc7ELcIf |

#### Вступление по приглашении

Запрос для получения доступа к альбому чужого пользователя по коду приглашению.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/invitation/**{code}**/join  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 404, 409 |

#### Удаление кода приглашения

Запрос для удаления кода приглашения.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/invitation/**{code}**  **Method:** DELETE | **Status:** 204 No Content  **Возможные статус-коды:** 204, 403, 404 |

#### Отзыв доступа для пользователя

Запрос для удаления доступа к своему альбому другого пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{album}**/accesses/**{user}**  **Method:** DELETE | **Status:** 204 No Content  **Возможные статус-коды:** 204, 403, 404 |

#### Отзыв доступа на себе

Запрос для удаления доступа к чужому альбому от себя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{album}**/accesses  **Method:** DELETE | **Status:** 204 No Content  **Возможные статус-коды:** 204, 403, 404 |

### Работа с играми

#### Загрузка картинок в альбом

Запрос для загрузки на ФС сервера картинок и информации об каждой в БД

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**/pictures  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** multipart/form-data  **Body:** {  "pictures[]": **{pictures}**,  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "successful": [  {  "id": 1,  "name": "test.jpg",  "date": "2024-10-29 19:56:10",  "size": 1696738,  "width": 1360,  "height": 2128,  “hash”: "df59b7d94c03b59a”  }, { ... }  ],  "errored": [  {  "name": "image.png",  "message": “already exist"  }, { ... }  ]  }  **Возможные статус-коды:** 200, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** |
| 1 | pictures | array | да | мин: 1 | Массив картинок |
| 1 | pictures.\* | file | нет | mimes: jpeg,png,gif | Картинка |

#### Получение списка картинок

Запрос для получения списка всех картинок в указанном альбоме. Так же выводится временная (на сутки) сигнатура доступа для получения превью/оригинала/загрузки.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**/pictures  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "sign": <**сигнатура**>  "pictures": [  {  "id": 1,  "name": "test.jpg",  "date": "2024-10-29 19:56:10",  "size": 1696738,  "width": 1360,  "height": 2128,  “hash”: "df59b7d94c03b59a”  },  { ... }  ],  }  **Возможные статус-коды:** 200, 401, 403, 404 |

#### Отображение превью картинки

Запрос для получения файла-превью указанного размера (разрешённый размер из публичной информации о сервере). Для аутентификация используется сигнатура доступа, которую можно получить из запроса списка картинок. Если указать запрещённый размер, то будет перенаправление на приближённый размер.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  /thumb/**{orient}{size}**  ?sign=**{sign}**  **Method:** GET | **Status:** 200 OK  **Content-Type:** image/jpeg  **Возможные статус-коды:** 200, 301, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | pictureId | integer | да | Код картинки | 529 |
|  | orient | "w"|"h"|"q" |  | Ось указания размера (по ширине, высоте или обрезка в квадрат) |  |
| 2 | size | integer | да | Размер по оси (пиксели) | 720 |
| 3 | sign | string | да | Сигнатура доступа | 2\_5EyJHc3aBoY1Ju |

#### Отображение картинки (оригинал)

Запрос для отображения файла. Для аутентификация используется сигнатура доступа, которую можно получить из запроса списка картинок.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  /original  ?sign=**{sign}**  **Method:** GET | **Status:** 200 OK  **Content-Type:** image/<"jpeg"|"png"|...>  **Возможные статус-коды:** 200, 403, 404 |

#### Скачивание картинки

Запрос для скачивания файла. Для аутентификация используется сигнатура доступа, которую можно получить из запроса списка картинок.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  /download  ?sign=**{sign}**  **Method:** GET | **Status:** 200 OK  **Content-Type:** image/<"jpeg"|"png"|...>  **Content-Disposition:**  attachment; filename=<имя\_файла>  **Возможные статус-коды:** 200, 403, 404 |

#### Удаление картинки

Запрос для удаления картинки. Удаляет связанные данные в БД и картинку в ФС.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  **Method:** DELETE  **Authorization:** Bearer **{token}** | Успех  **Status:** 204 Deleted  **Возможные статус-коды:** 204, 401, 403, 404 |

### Работа с библиотекой

#### Создание личного тега

Запрос для создания личного тега пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "value": **{value}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "id": <**id\_тега**>,  "name": <**название**>  }  **Возможные статус-коды:** 201, 401, 409 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | value | string | да | макс: 255, уник(для юзера) | Название | Семья |

#### Получение списка личных тегов

Запрос на получение списка личных тегов.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** [  {  "id": <**id\_тега**>,  "name": <**название**>  },  { ... },  ]  **Возможные статус-коды:** 204, 401 |

#### Добавление тега к картинке

Запрос для добавления связи тега с картинкой.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /pictures/**{pictureId}**  /tags/**{tagId}**  **Method:** POST  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | tagId | string | да | существовать | Код тега | 5 |

#### Удаление тега с картинки

Запрос для удаления связи тега с картинкой.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /pictures/**{pictureId}**  /tags/**{tagId}**  **Method:** DELETE  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

### Работа с профилем

#### Создание жалобы на альбом

Запрос для создания жалобы на альбом.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /complaint  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "description": **{description}**,  "typeId": **{typeId}**  } | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

#### Создание жалобы на картинку

Запрос для добавления жалобы на картинку.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /pictures/**{pictureId}**  /complaint  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "description": **{description}**,  "typeId": **{typeId}**  } | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** |
| 1 | description | string | да | макс: 255 | Пояснение |
| 2 | typeId | integer | да | существовать | Тип жалобы |

#### Получение списка жалоб

Запрос для получения списка жалоб. Администратору выведутся все, пользователю — жалобы который он создал.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/complaints  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** [  {  "id": <**id\_жалобы**>,  "description": <**пояснение**>,  "from\_user": <**данные\_жалующегося**>,  "about\_user": <**данные\_на\_кого\_жалоба**>,  <"album"|"picture">: {  <**данные\_предмета\_жалобы**>  }  }, { ... }  ]  **Возможные статус-коды:** 200, 401, 403 |

### Работа с облачными сервисами

#### Создание предупреждения

Запрос для добавления предупреждения на пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/**{user}**/warnings  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "comment": **{description}**  } | **Status:** 201 Created  **Возможные статус-коды:** 201, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** |
| 1 | comment | string | нет | Пояснение |

#### Удаление предупреждения

Запрос для удаления предупреждения с пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/**{user}**/warnings/**{warnings}**  **Method:** DELETE  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404 |

### 

### Работа с пользователями

#### Получение себя

Запрос на данных о себе.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/me  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 1,  "name": "Testello",  "login": "test",  <"role": "admin">  }  }  **Возможные статус-коды:** 200, 401 |

#### Редактирование себя

Запрос на изменение данных о себе.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/me  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "name": **{nickname}**,  "login": **{login}**,  "password": **{password}**  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 1,  "name": "Testello",  "login": "test",  <"role": "admin">  }  }  **Возможные статус-коды:** 200, 401, 422 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | nickname | string | нет | мин: 2, макс: 255 | Имя юзера | Testello2 |
| 2 | login | string | нет | мин: 2, макс: 64, уник,  regex: ^[a-zA-Z0-9\_-]+$ | Логин | test2 |
| 3 | password | string | нет | мин: 8, макс: 255 | Пароль | NewPass! |

#### Просмотр списка пользователей

Запрос на просмотр списка пользователей.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "users": [  {  "id": 2,  "name": "Чел",  "isBanned": false,  "login": "test1",  <"role": "admin">  "complaintsAboutCount": 21,  "complaintsAboutAcceptedCount": 0,  "complaintsFromCount": 18,  "complaintsFromAcceptedCount": 9,  "albumsCount": 1,  "albumsViaAccessCount": 2,  "picturesCount": 31  }, { ... }, { ... }  ]  }  **Возможные статус-коды:** 200, 401, 403 |

#### Просмотр пользователя

Запрос просмотра пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/**{userId}**  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 2,  "name": "Чел",  "isBanned": false,  "login": "test1",  <"role": "admin">  "complaintsAbout": [  {  "id": 1,  "status": null,  "description": "Плохие картинки"  }, { ... }, { ... }  ],  "complaintsFrom": [  {  "id": 1,  "status": null,  "description": "Мне не нравится"  }, { ... }, { ... }  ],  "albumsCount": 1,  "albumsViaAccessCount": 2,  "picturesCount": 31,  "warnings": [  {  "id": 1,  "comment": "Больше так не делай"  }, { ... }, { ... }  ],  }  }  **Возможные статус-коды:** 200, 401, 403, 404 |

#### Редактирование пользователя

Запрос редактирование пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/**{userId}**  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "name": **{nickname}**,  "login": **{login}**,  "password": **{password},**  "role\_id": **{roleId}**,  "is\_banned": **{**isBanned**},**  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 2,  "name": "Чел",  "isBanned": false,  "login": "test1",  <"role": "admin">  }  }  **Возможные статус-коды:** 200, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | roleId | integer | нет | существовать | ИД роль | 2 |
| 2 | isBanned | boolean | нет |  | Статус блокировки | true |

### Работа с настройками

#### Просмотр публичных настроек

Запрос публичных настроек.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/  **Method:** GET | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "settings": {  "allowed\_upload\_mimes": [  "jpeg", "jpg", "png", "gif"  ],  "allowed\_preview\_sizes": [  144, 240, 360, 480, 720, 1080  ],  "warning\_limit\_for\_ban": 3,  "free\_storage\_limit": 5368709120,  "complaint\_types": [  {  "id": 5,  "name": "Нарушение авторских прав"  },  {  "id": 6,  "name": "Шок-контент"  }  ],  "is\_upload\_disabled": true  }  }  **Возможные статус-коды:** 200 |

## Требования к десктопному приложению клиента | Схематические макеты интерфейса

В рамках данного проекта используются схематические макеты (wireframe), определяющие расположение элементов интерфейса и логику пользовательского взаимодействия. Дизайн и стилизация не проработаны на данном этапе.

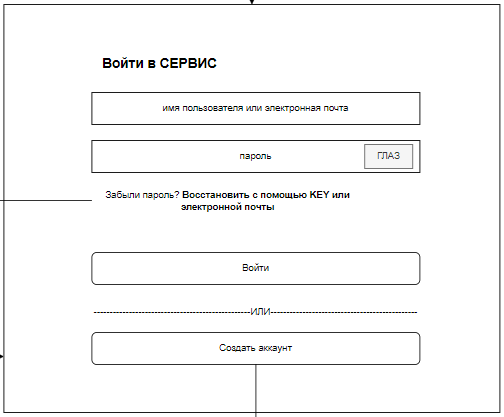


Рисунок 10 – Экран авторизации

Описание логики взаимодействия экрана авторизации (см. Рисунок 10)

1. Основной функционал экрана

Этот экран предназначен для входа пользователя в сервис. Он содержит стандартные элементы формы авторизации.

2. Размещение элементов и их функциональность

Заголовок: "Войти в СЕРВИС" – указывает на назначение экрана.

Поле ввода "имя пользователя или электронная почта":

Пользователь вводит либо свой логин, либо email.

Валидация проверяет корректность введенных данных.

Поле ввода "пароль":

Пользователь вводит пароль.

Справа от поля есть кнопка с иконкой "ГЛАЗ" – предназначена для показа/скрытия пароля.

Ссылка "Забыли пароль?":

Если пользователь забыл пароль, он может нажать на эту ссылку.

Открывается процесс восстановления (например, через KEY или email).

Кнопка "Войти":

После ввода данных пользователь нажимает на кнопку для авторизации.

Если данные верны – происходит вход.

Если данные неверны – показывается сообщение об ошибке.

Разделитель "ИЛИ":

Визуально отделяет процесс входа от возможности регистрации.

Кнопка "Создать аккаунт":

Если у пользователя нет учетной записи, он может зарегистрироваться.

При нажатии переходит на экран регистрации.

3. Возможные дополнительные механики

Автозаполнение полей (если уже входили ранее).

Подсветка полей при ошибке (например, красная рамка при неправильном вводе).

Поддержка клавиши Enter для быстрого входа.

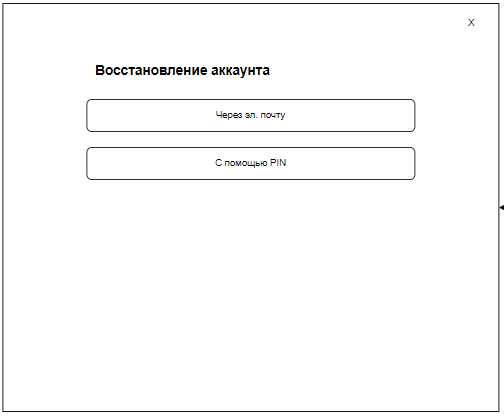


Рисунок 11 – Экран восстановление аккаунта

Описание логики взаимодействия экрана восстановления аккаунта (см. Рисунок 11)

1. Основное назначение экрана

Этот экран предназначен для восстановления доступа к учетной записи. Пользователь может выбрать один из двух способов восстановления.

2. Размещение элементов и их функциональность

Заголовок "Восстановление аккаунта"

Объясняет цель экрана.

Кнопка "Через эл. почту"

Если пользователь выбирает этот вариант, открывается экран, где он вводит свою электронную почту.

На почту отправляется код или ссылка для сброса пароля.

Кнопка "С помощью PIN"

Если у пользователя есть резервный PIN-код, он может выбрать этот вариант.

Открывается экран ввода PIN-кода для подтверждения личности.

3. Возможные дополнительные механики

Если у пользователя нет доступа к почте или он не помнит PIN, может быть предусмотрена альтернативная помощь (например, обращение в поддержку).

В случае неверного ввода PIN или email можно показывать предупреждения.

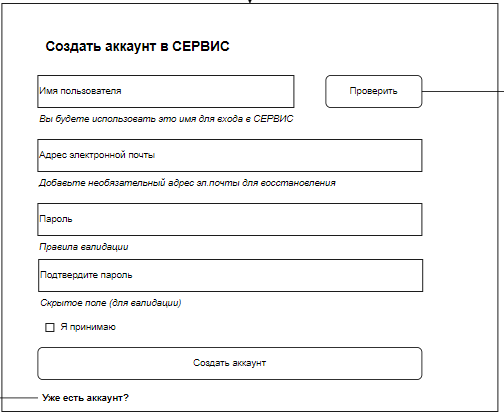


Рисунок 12 – Экран создания аккаунта

Описание логики взаимодействия экрана регистрации (см. Рисунок 12)

1. Основное назначение экрана

Этот экран позволяет пользователю создать новый аккаунт в сервисе. Он включает в себя ввод имени пользователя, электронной почты, пароля и принятие условий.

2. Размещение элементов и их функциональность

Заголовок "Создать аккаунт в СЕРВИС"

Объясняет цель экрана.

Поле "Имя пользователя"

Пользователь вводит уникальное имя для входа.

Справа кнопка "Проверить" – отправляет введенное имя на проверку доступности.

Текст-подсказка под полем имени

Оповещает, что имя будет использоваться для входа.

Поле "Адрес электронной почты"

Ввод основного email, на который придет подтверждение регистрации.

Дополнительное поле "Адрес эл. почты для восстановления"

Опциональное поле, помогает при утере доступа к основному email.

Поле "Пароль"

Пользователь вводит пароль.

Под полем есть подсказка "Правила валидации" – возможно, при наведении или клике отображаются требования к паролю.

Поле "Подтвердите пароль"

Пользователь повторно вводит пароль для проверки.

Скрытое поле (для валидации)

Возможно, используется для защиты от ботов.

Чекбокс "Я принимаю"

Без отметки кнопка "Создать аккаунт" должна быть неактивной.

Кнопка "Создать аккаунт"

После заполнения всех полей пользователь нажимает для регистрации.

При успешной регистрации может быть редирект на страницу входа.

Если есть ошибки (например, имя занято, пароли не совпадают), появляются сообщения.

Ссылка "Уже есть аккаунт?"

Ведет на экран авторизации, если пользователь уже зарегистрирован.

3. Возможные дополнительные механики

Мгновенная проверка имени пользователя при вводе.

Подсказки и требования к паролю (например, "Минимум 8 символов, заглавные буквы, цифры").

Интерактивная кнопка "Создать аккаунт" – становится активной только после заполнения всех обязательных полей.

Капча или антибот-механизм в скрытом поле.

Подтверждение email после регистрации (отправка письма с кодом или ссылкой).

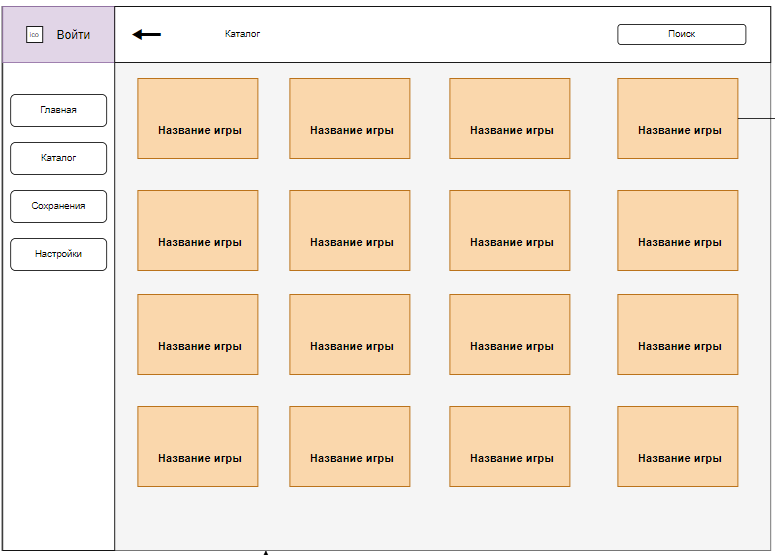


Рисунок 13 – Каталог

Описание логики взаимодействия экрана каталога игр (см. Рисунок 13)

1. Основное назначение экрана

Этот экран предоставляет пользователю каталог доступных игр. Отсюда можно выбрать игру для просмотра информации и управления сохранениями.

2. Размещение элементов и их функциональность

Боковое меню (слева):

Кнопка "Войти" – авторизация пользователя.

Главная, Каталог, Сохранения, Настройки – основные разделы сервиса.

Основная панель (справа):

Кнопка "Назад" (стрелка влево) – возвращает на предыдущий экран.

Заголовок "Каталог" – обозначает текущий раздел.

Поиск – позволяет искать игры по названию.

Список игр (сетка карточек):

Каждая карточка представляет игру с ее названием.

При клике на игру происходит переход на экран информации о игре.

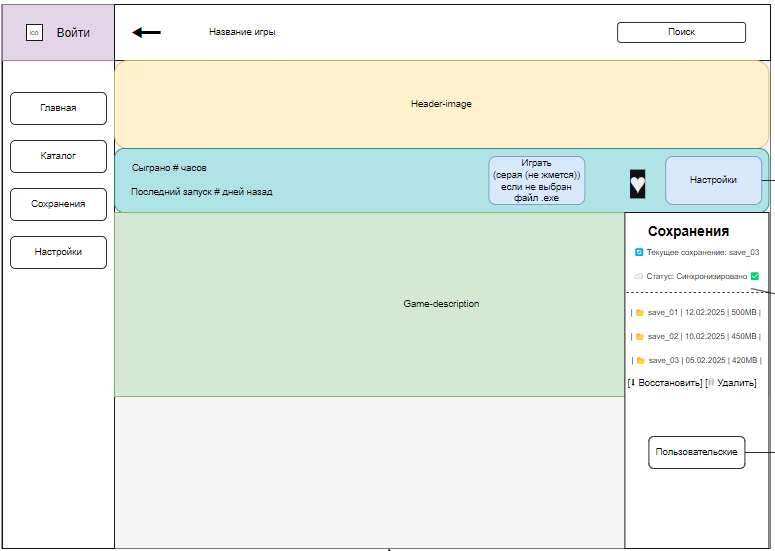


Рисунок 14 – Просмотр игры

Описание логики взаимодействия экрана информации о игре (см. Рисунок 14)

1. Основное назначение экрана

Этот экран предоставляет детальную информацию об игре, позволяет запустить её, управлять настройками и сохранениями.

2. Размещение элементов и их функциональность

Боковое меню – аналогично экрану каталога.

Основная панель:

Кнопка "Назад" – возвращает в каталог.

Название игры – отображается в заголовке.

Header-image – изображение игры.

Информационный блок (синяя область):

Сыграно # часов – показывает статистику времени в игре.

Последний запуск # дней назад – информация о недавнем запуске.

Кнопка "Играть" – запускает игру (если .exe-файл не выбран, кнопка неактивна).

Кнопка "Настройки" – открывает экран настроек игры.

Список сохранений (справа):

Отображает текущее сохранение и его статус.

Список сохранений с датой, размером файла и кнопками:

"[Восстановить]" – загружает сохранение.

"[Удалить]" – удаляет сохранение.

"Пользовательские" – возможно, раздел для ручных сохранений.

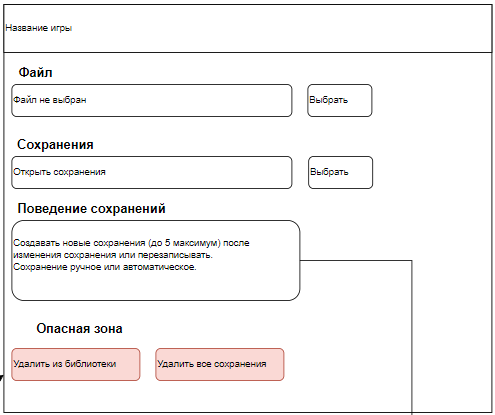


Рисунок 15 – Настройки игры

Описание логики взаимодействия экрана настроек игры (см. Рисунок 15)

1. Основное назначение экрана

Этот экран позволяет управлять настройками игры, файлами сохранений и удалением игры из библиотеки.

2. Размещение элементов и их функциональность

Название игры – отображается в заголовке.

Файл

Поле "Файл не выбран" + кнопка "Выбрать" – позволяет указать .exe-файл для запуска игры.

Сохранения

Поле "Открыть сохранения" + кнопка "Выбрать" – настройка пути к сохранениям игры.

Поведение сохранений

Текстовое описание параметров:

Можно сохранять вручную или автоматически.

Можно настроить перезапись или создание новых сохранений (до 5 максимум).

Опасная зона

"Удалить из библиотеки" – удаляет игру из списка, но не файлы.

"Удалить все сохранения" – полное удаление всех сохранений.

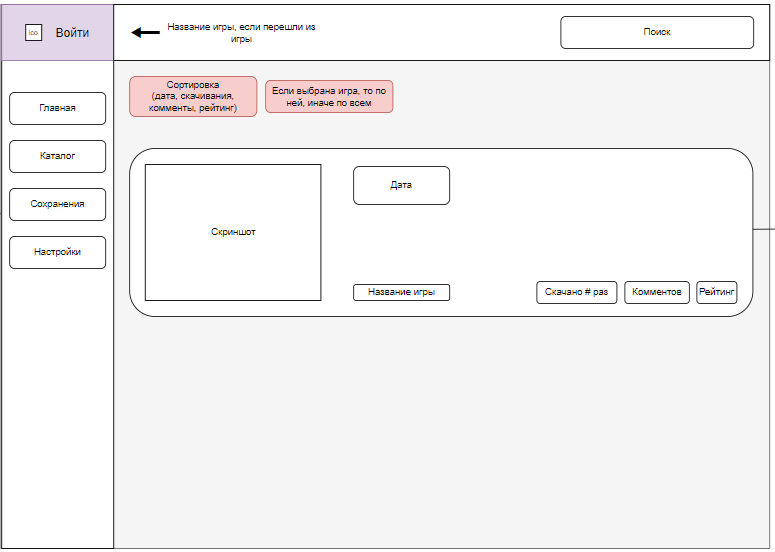


Рисунок 16 – Сохранения

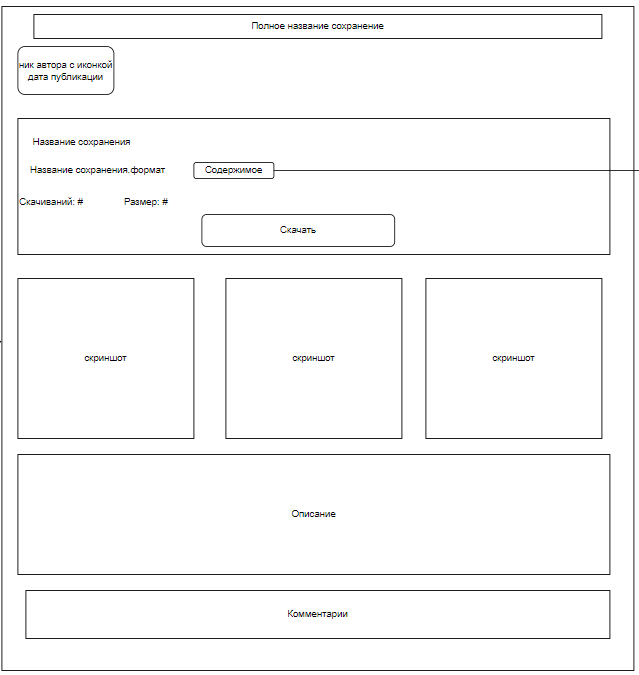


Рисунок 17 - Просмотр сохранения

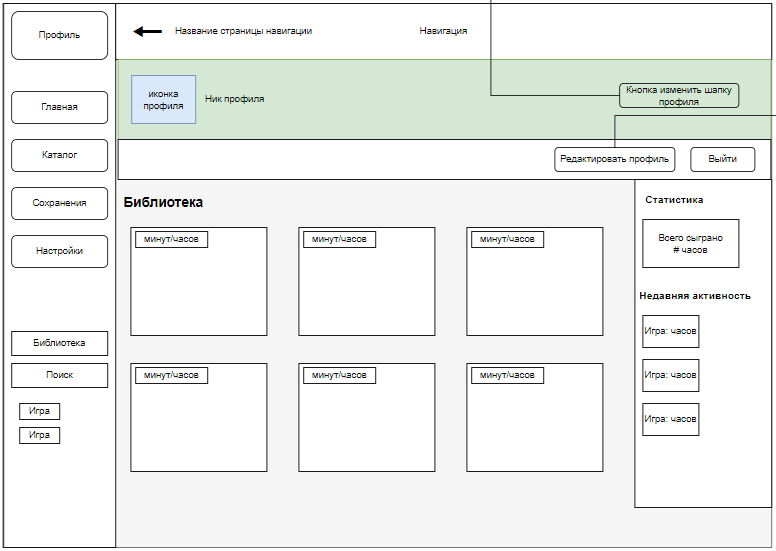


Рисунок 18 - Просмотр профиля

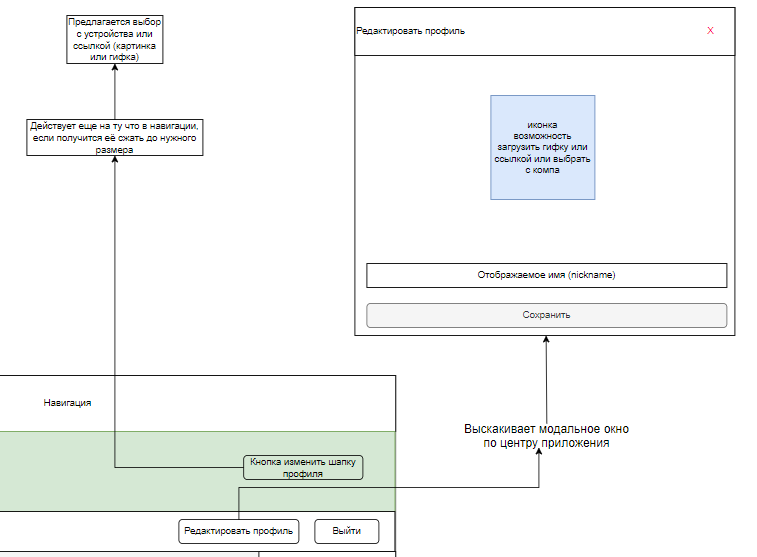


Рисунок 19 - Что-то ещё

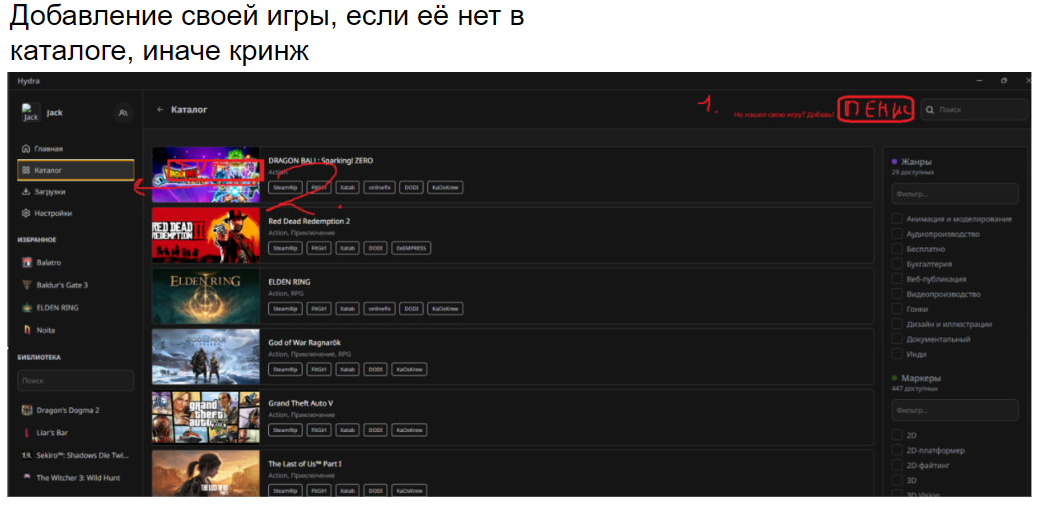


Рисунок 20 - Добавление своей игры

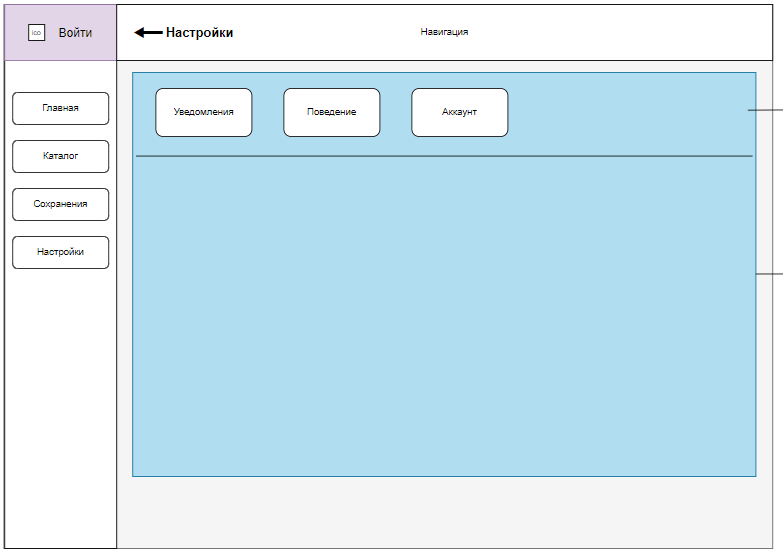


Рисунок 21 - Настройки из навигации

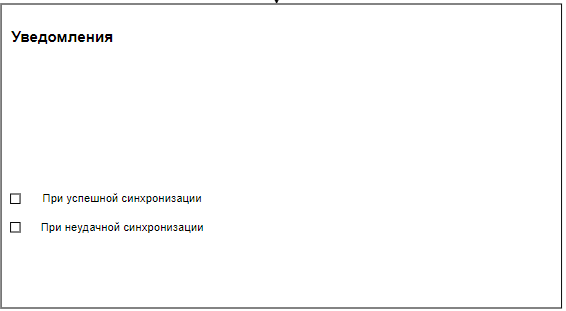


Рисунок 22 - Уведомления в настройках

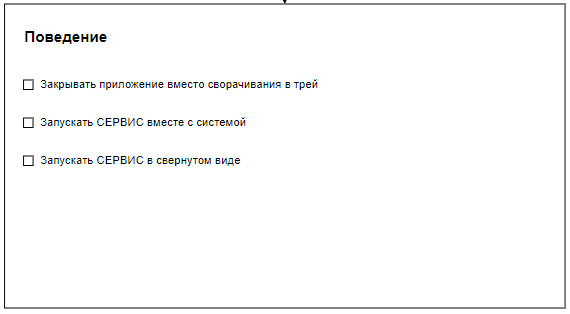


Рисунок 23 - Поведение в настройках

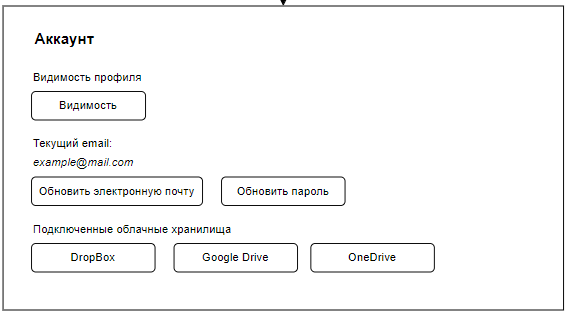


Рисунок 24 - Настройки аккаунта

# Глава 2. Проектирование и разработка базы данных

## Концептуальная модель данных

Концептуальная модель — это формализованное описание предметной области, не связанное с какими-либо компьютерными средствами. Сущности (таблицы) обозначаются прямоугольниками; Атрибуты (поля таблиц) — овалы, которые могут быть объедены в скруглённые прямоугольники; Связи с обозначением мощности — стрелками (в направлении, где мощность равна многим — двойная стрелка, а со стороны, где она равна единице — одинарная).

Анализируя предметную область, можно выделить следующие сущности с их атрибутами (см. Рисунок 25):

1. Сущность «**Роль**» с атрибутом «Название».
2. Сущность «**Предупреждение**» с атрибутом «Пояснение».
3. Сущность «**Пользователь**» с атрибутами «Логин», «Никнейм», «Хеш пароля», «Количество жалоб», «Состояние бана».
4. Сущность «**Тег**» с атрибутом «Название».
5. Сущность «**Доступ к альбому**», связывающая сущности «Альбом» и «Пользователь».
6. Сущность «**Токен**», c атрибутом «Содержимое».
7. Сущность «**Альбом**» с атрибутами «Название», «Путь на устройстве».
8. Сущность «**Код** приглашения» с атрибутами «Значение», «Время жизни».
9. Сущность «**Картинка**» с атрибутами «Название», «Хеш», «Размер», «Ширина», «Высота», «Дата».
10. Сущность «**Жалоба**» с атрибутом «Пояснение».
11. Сущность «**Тип жалобы**», c атрибутом «Содержимое».
12. Сущность «**Тег-Картинка**», связывающая сущности «Тег» и «Картинка».

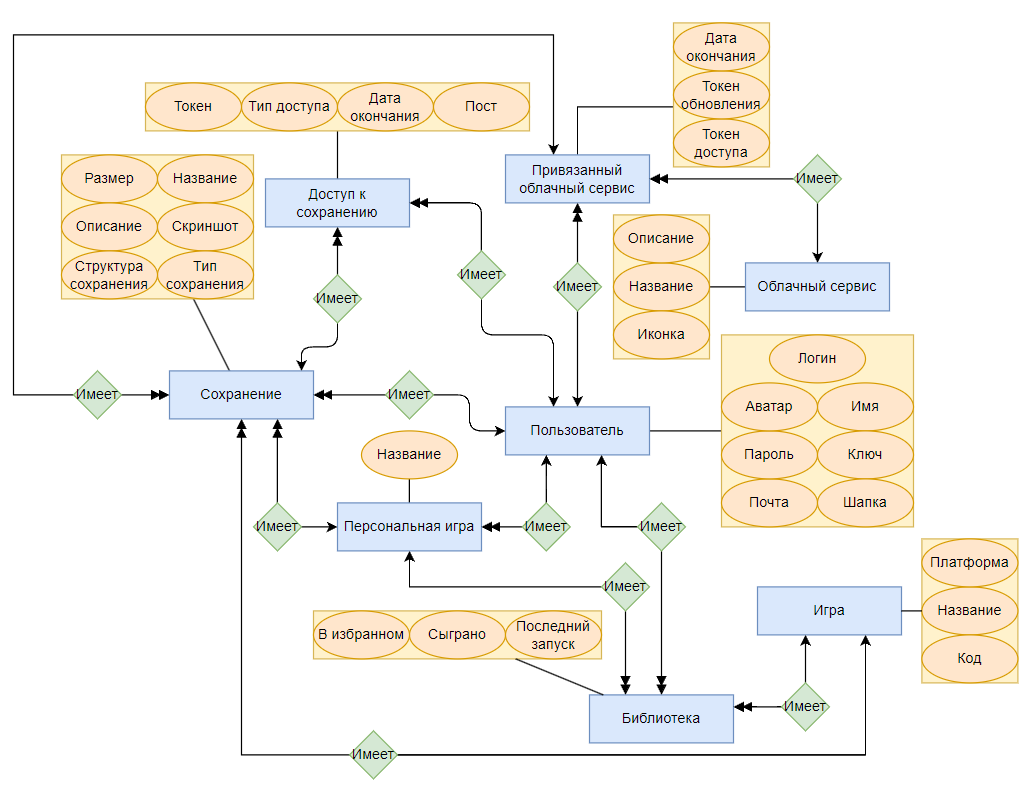


Рисунок 25 – Концептуальная модель данных

## Логическая модель данных

Логическая модель базы данных — схема базы данных, выраженная в понятиях модели данных. Этим отличается от концептуальной модели, описывающей семантику предметной области без указания технологии (конкретных методов реализации), и от физической модели, которая описывает конкретные физические механизмы, применяемые для хранения данных в накопителях.

Используя нашу концептуальную модель данных и выполнив нормализацию данных до 3 нормальной формы включительно, построим логическую модель данных (см. Рисунок 26):



Рисунок 26 – Логическая модель данных

## Физическая модель данных

Физическая модель данных — это модель данных, описанная с помощью средств конкретной системы управления базами данных (СУБД). Физическая модель данных строится на базе логической путем добавления особенностей конкретной СУБД. К таким особенностям могут относиться поддерживаемые СУБД типы данных, соглашения о присвоении имен таблицам, атрибутам и т.д. Физическая модель данных фактически является готовым заданием на создание БД, имея которое можно реализовать БД в выбранной СУБД.

Выбранной СУБД является MySQL. Используя логическую модель, построим физическую модель данных (см. Рисунок 27):

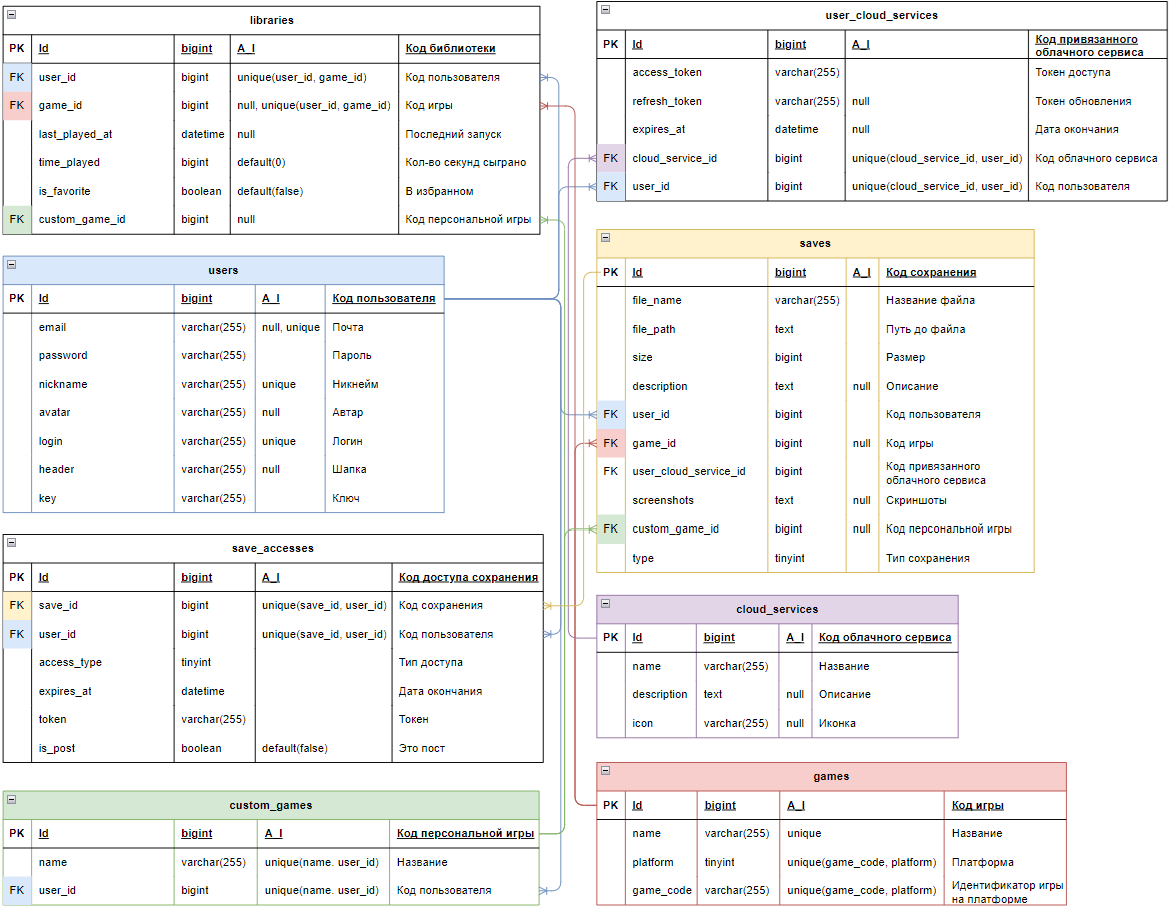


Рисунок 27 – Физическая модель данных